EDGEWOOD
RESEARCH, DEVELOPMENT & ENGINEERING CENTER
U.S. ARMY CHEMICAL AND BIOLOGICAL DEFENSE COMMAND

ERDEC-TR-212

COMPUTATION OF ELECTROMAGNETIC SCATTERING PARAMETERS FOR LOGNORMAL DISTRIBUTIONS OF MAGNETIC SPHERES: THEORY AND ALGORITHMS

Merrill E. Milham

RESEARCH AND TECHNOLOGY DIRECTORATE

October 1994

Aberdeen Proving Ground, MD 21010-5423

Disclaimer

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorizing documents.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	1994 October	Final, 91 Jan - 94 Jan

4. TITLE AND SUBTITLE
Computation of Electromagnetic Scattering Parameters for Lognormal Distributions of Magnetic Spheres: Theory and Algorithms

5. FUNDING NUMBERS
PR-1O162622A552

6. AUTHOR(S)
Milham, Merrill E.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
DIR, ERDEC,* ATTN: SCBRD-RTE, APG, MD 21010-5423

8. PERFORMING ORGANIZATION REPORT NUMBER
ERDEC-TR-212

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
*When this study was conducted, ERDEC was known as the U.S. Army Chemical Research, Development and Engineering Center, and the author was assigned to the Research Directorate.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

In this report, relevant parts of the scattering theory for magnetic spheres are presented. Mass extinction coefficients, and the lognormal size distribution are defined. The theory and algorithms for integrating scattering parameters over size distributions are developed. The integrations are carried out in terms of dimensionless scattering, and size distribution parameters, which are simply related to the usual mass scattering coefficients. Fortran codes, which implement the algorithmic design, are presented, and examples of code use are given. Code listings are included.

14. SUBJECT TERMS		15. NUMBER OF PAGES
Magnetic sphere	Electromagnetic scattering	32
Complex permeability	Mass extinction coefficients	16. PRICE CODE
Complex permittivity	(Continued on page 2)	

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

14. SUBJECT TERMS (Continued)

Lognormal distributions
Dimensionless extinction coefficients

PREFACE

The work described in this report was authorized under Project No. 1O162622A552, Smoke Technology. This work was started in January 1991 and completed in January 1994.

The use of trade names or manufacturers' names in this report does not constitute an official endorsement of any commercial products. This report may not be cited for purposes of advertisement.

This report has been approved for release to the public. Registered users should request additional copies from the Defense Technical Information Center; unregistered users should direct such requests to the National Technical Information Service.

Blank

CONTENTS

Blank

COMPUTATION OF ELECTROMAGNETIC SCATTERING PARAMETERS FOR LOGNORMAL DISTRIBUTIONS OF MAGNETIC SPHERES: THEORY AND ALGORITHMS

1. INTRODUCTION

A previous report[1] described the theory, algorithmic design, and testing of a subroutine for computing the electromagnetic scattering from a magnetic sphere. In this report the methodology for using this subroutine to compute the mass extinction, scattering, absorption, and backscatter coefficients for lognormally distributed particulate ensembles is developed.

In what follows the relevant parts of Mie scattering theory are presented. Mass extinction coefficients, and the lognormal size distribution are defined, and the theory and algorthims for integrating Mie scattering parameters over size distributions are developed. The integrations are carried out in terms of dimensionless scattering, and size distribution parameters, which are simply related to the usual mass scattering coefficients.

2. LIGHT SCATTERING AND CLOUD MACROPHYSICS

For spheres, the efficiencies, the ratio of the optical cross section to the geometric cross section, for extinction (Q_e), scattering (Q_s), absorption (Q_a), and backscatter (Q_b) are:[2]

$$Q_e = \frac{2}{X^2} \sum_{n=1}^{\infty} (2n+1)\,\mathrm{Re}\,(a_n + b_n) \tag{1}$$

$$Q_s = \frac{2}{X^2} \sum_{n=1}^{\infty} (2n+1)[\,|a_n|^2 + |b_n|^2\,] \tag{2}$$

$$Q_a = Q_e - Q_s \tag{3}$$

$$Q_b = \frac{1}{X^2} \left| \sum_{n=1}^{\infty} (-1)^n (2n+1)(a_n - b_n) \right| \tag{4}$$

where

$$a_n = \frac{\sqrt{\tilde{\mu}}\, J_{n+1/2}(X) J'_{n+1/2}(mX) - \sqrt{\tilde{\epsilon}}\, J_{n+1/2}(mX) J'_{n+1/2}(X)}{\sqrt{\tilde{\mu}}\, H^{(2)}_{n+1/2}(X) J'_{n+1/2}(mX) - \sqrt{\tilde{\epsilon}}\, J_{n+1/2}(mX) H^{(2)\prime}_{n+1/2}(X)} \tag{5}$$

$$b_n = \frac{\sqrt{\tilde{\epsilon}}\, J_{n+1/2}(X) J'_{n+1/2}(mX) - \sqrt{\tilde{\mu}}\, J_{n+1/2}(mX) J'_{n+1/2}(X)}{\sqrt{\tilde{\epsilon}}\, H^{(2)}_{n+1/2}(X) J'_{n+1/2}(mX) - \sqrt{\tilde{\mu}}\, J_{n+1/2}(mX) H^{(2)\prime}_{n+1/2}(X)} \tag{6}$$

$J_{n+1/2}(z)$, $H_{n+1/2}(z)$ are the Bessel functions and Hankel functions (second kind) of half-integer order.[3] $X = \pi D/\lambda$ is the size parameter where, D, is the diameter in μm; λ, the wavelength in μm; and $m = \sqrt{\tilde{\mu}\tilde{\epsilon}}$, the refractive index for which $\tilde{\mu}$, the permeability, and $\tilde{\epsilon}$, the permittivity, are complex quantities. Primes denote derivatives with respect to the argument.

For ensembles of spherical particles, the mass extinction, scattering, absorption, or backscatter coefficients (in square meters per gram) at wavelength λ are defined as

$$\alpha = \sum_i A_i \,/\, \sum_i m_i \tag{7}$$

where A_i is the optical cross section for extinction, scattering, absorption, or backscatter of the *i*th particle (in square meters), m_i is the mass of the *i*th particle (in grams), and the sum extends over the ensemble of particles contained in the optical path. For a continuous distribution of particles, the expression for α becomes

$$\alpha = \int \alpha(D)\, dm \tag{8}$$

$\alpha(D)$ is the coefficient of extinction, scattering, absorption, or backscatter for a particle of size D, $= 3Q/2\rho D$ where Q is given by the expressions for Q_e, Q_s, Q_a, and Q_b (Equations 1-4); by convention an additional factor of 4π is included in the denominator of this expression when $Q = Q_b$.[4] dm is the mass-distribution function which for the purposes of this work is chosen to be the lognormal distribution function:

$$dm = (2\pi \ln^2 \sigma_g)^{-1/2} \exp\{-\tfrac{1}{2}[\ln(D/D_m)/\ln \sigma_g]^2\}\, d(\ln D) \tag{9}$$

3. NUMERICAL METHODS - ALGORITHM DESIGN

Evaluation of the indicated quadratures over lognormal particle-size distributions can be a difficult numerical problem. The determination of the Q's is a computationally intense procedure that must be carried out over a fine integration mesh to ensure accurate results; this is especially true for Q_e with no absorption ($\mathrm{Im}(\tilde{m}) = 0$) or Q_b which have substantial resonant structure.

Because the Q's are evaluated in terms of the dimensionless size parameter, X, it is convenient to formulate the quadrature problem in terms of dimensionless quantities. The α's can be put in a dimensionless form as follows:

$$\alpha_d = \alpha \cdot \lambda \cdot \rho \tag{10}$$

Where ρ is the density in g cm^{-3}. Also, converting D, D_m in the particle size distribution to X, X_m leaves the functional form of the distribution unchanged.

After converting these quantities to dimensionless form the integrations are carried out by establishing minimum (X_o) and maximum (X_l) size parameters that are the end points for N evenly spaced abscissae on a logarithmic grid and then applying an appropriate numerical integration scheme such as Simpson's rule. The X grid is given by

$$X_0,\ RX_0,\ \cdots,\ R^{N-1}X_0 \tag{11}$$

where $X_o = X_m/\sigma_g^n$, $X_l = X_m\sigma_g^n = R^{N-1}\,X_o$, and $R = \Delta \ln X = \sigma_g^{2n/(N-1)}$. n, the number of geometric standard deviations, is chosen to be three or greater so that regions outside the grid contribute a numerically negligible amount to the integral. The dimensionless coefficients are then given by the sum

$$\alpha_d \approx (9\pi/8\ln^2\sigma_g)^{1/2} \sum_{j=1}^{N} X_j\, Q \exp\{-1/2\,[(\ln X_j - \ln X_m)/\ln\sigma_g]^2\}\,\Delta(\ln X) \tag{12}$$

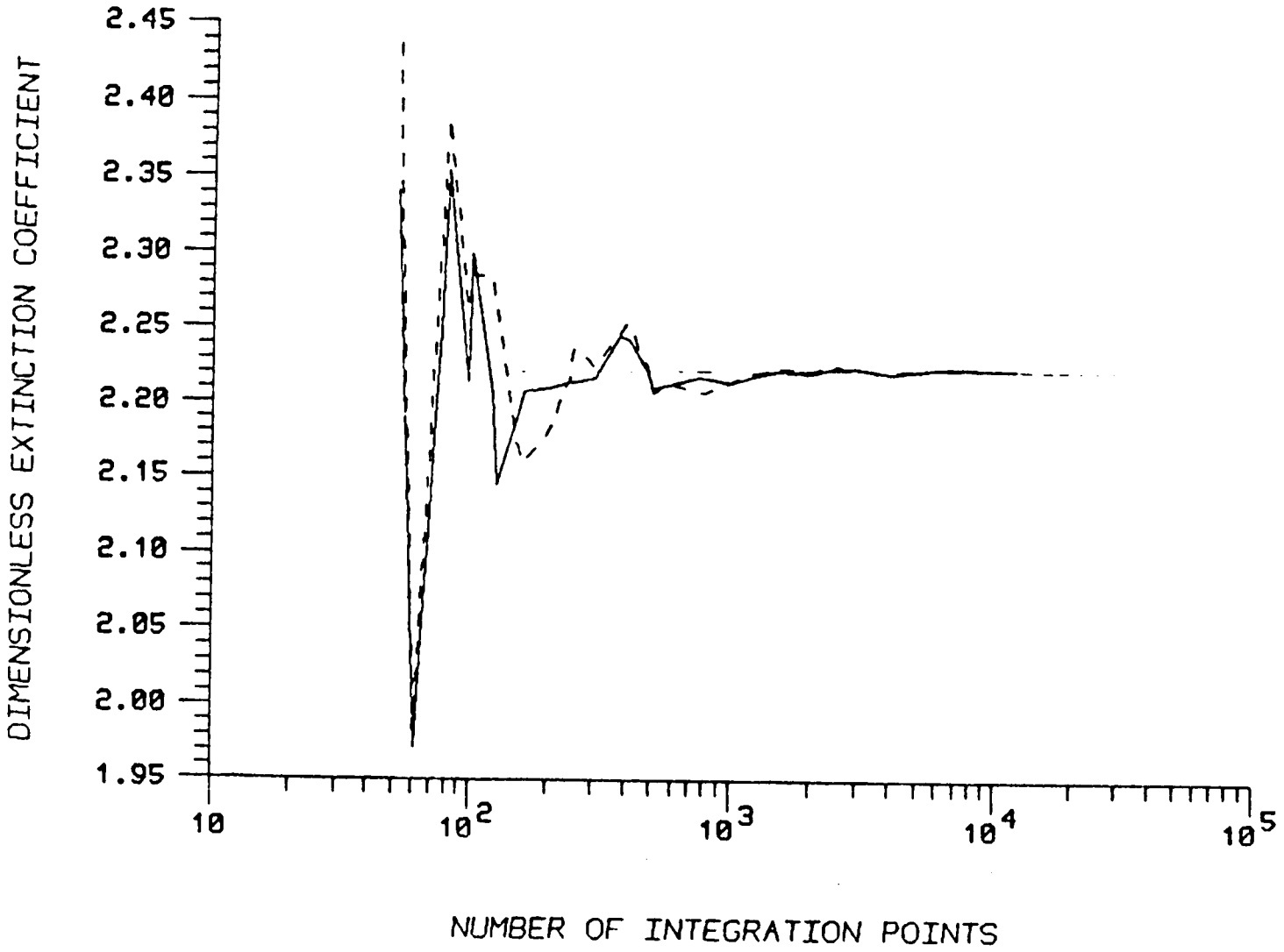
When $Q = Q_b$ then the expression for α_d must be multiplied by an additional factor of $1/4\pi$. The Figure shows the effect of varying the number of terms in the sum in Equation 12 for the dimensionless extinction coefficient for m = (3.0, 0.); μ = (1.0, 0.).

4. RESULTS AND DISCUSSION

The listings of a main (driver) program (disint), which computes either dimensioned or dimensionless extinction, scattering, absorption, and backscatter coefficients is displayed in Appendix A along with the listings for the auxiliary routines: dxtcs, magsph, zbjy, and simp. The Bessel function subroutine package by Amos[5] is required to implement these routines.

Appendix B lists the input to disint and then displays the resulting output for three sample calculations. The first two sample problems compute the dimensionless extinction coefficients for a mass median size parameter of 10, a refractive index of (1.5, -0.1). The first example uses a very narrow distribution (σ_g = 1.01); the results of this calculation can be inverted by the relationship,

$$Q \approx \frac{2X_m}{3\pi}\,\alpha \tag{13}$$



$N_p = 12001$

$n = 3.$

$k = 0.$

$X_m = 10.$

$\sigma_g = 3.0$

and the results compared with the calculation for monodisperse particle calculation of Wiscombe[6] (Table). The next example gives the results for a particle with same refractive index and a geometric standard deviation of 1.4. The final example is a sample problem that exercises the code for the case of magnetic spheres with $\mu = \epsilon = (2.24, -.3)$, which are lognormally distributed with an MMD of 10 μm and a σ_g of 1.5. It is known that the backscatter vanishes for spheres with these material properties;[1-7] this sample problem illustrates this point.

Table. Efficiency Factor Comparisons

Distribution	Q_e	Q_s	Q_a
Monodisperse	2.4598	1.2351	1.2246
Lognormal[a]	2.4593	1.2346	1.2246

[a] $\sigma_g = 1.01$

Blank

LITERATURE CITED

1. Milham, Merrill E., Electromagnetic Scattering by Magnetic Spheres: Theory and Algorithms, ERDEC-TR-207, U.S. Army Edgewood Research, Development and Engineering Center, MD, October 1994, UNCLASSIFIED Report.

2. Van de Hulst, H.C., Light Scattering by Small Particles, Dover, NY, 1981.

3. Abramowitz, M., and Stegun, I.A., Handbook of Mathematical Functions, GPO, Washington, DC, 1964.

4. Bohren, C.F., and Huffman, D.R., Absorption and Scattering of Light by Small Particles, Wiley, New York, NY, 1983.

5. Amos, D.E., "Algorithm 644: A Portable Package for Bessel Functions of a Complex Argument and Nonnegative Order," ACM Trans Math Software Vol 12, pp 265-273, 1986.

6. Wiscombe, W.J., Mie Scattering Calculations: Advances in Technique and Fast, Vector-Speed Computer Codes, NCAR/TN-140+STR, National Center for Atmospheric Research, Boulder, CO, 1979.

7. Kerker, M., Wang, D.-S., and Giles, C.L., "Electromagnetic Scattering by Magnetic Spheres," J. Opt. Soc. Am. Vol. 73, pp 765-767 (1983).

Blank

```
      program disint
c**********************************************************************
c
c   Program disint computes extinction, scattering, absorption, and
c   backscatter coefficients for lognormal distributions of homogeneous
c   magnetic or nonmagnetic spheres.
c
c         Merrill Milham       >>> version 1.0 <<<          JANUARY 1994
c
c   inputs:
c                       ALL INPUT IS IN LIST DIRECTED FORMAT
c
c          line #1
c                        mmd = mass median diameter (micron)    (real*8)
c                     sigmag = geometric standard deviation     (real*8)
c                        rho = density of the particulate material
c                              if rho .le. zero dimensionless
c                              extinction, scattering, absorption
c                              and backscatter coefficients are
c                              computed. (g cm-3)                (real*8)
c                       ndfp = number of logarithmically equally spaced
c                              abscissa values over which the various
c                              scattering coefficients are to be
c                              integrated                       (integer)
c                       nsig = number of sigmas on each side of the mmd over
c                              which the scattering coefficients are to be
c                              integrated.  No correction is made for excluded
c                              distribution tails.               (real*8)
c
c          line #2
c                        nwl = number of wavelengths and material
c                              properties to be read            (integer)
c                       flag = 'r' for refractive index data or
c                              'p' for permittivity data        (character*1)
c                         wl = wavelength, up to 10 values (micron) (real*8)
c                         mp = complex index or permittivity values
c                              to be read, up to 10 values      (complex*16)
c          line #3
c                        muu = complex permeability values to be read
c                              nwl values are required          (complex*16)
c
c   output-
c line 1: mmx or mmd (micron)  sigmag  density (g cm-3) ndfp  nsig
c line 2: wavelength (micron)  ext. coef. (m sq / g)  scat. coef. (m sq / g)
c line 3:                      abs. coef. (m sq / g)  bsca. coef. (m sq/g/sr)
c line 4: permittivity  permeability
c line 5: refractive index
c
c            If dimensionless quantities are computed, the output is so
c            labeled.
c
c   subroutines used:
```

```
c
c                dxtcs - returns integrated extinction coefficients for a
c                        homogeneous magnetic sphere or homogeneous
c                        nonmagnetic sphere if the permeability = (1,0)
c
c*************************************************************************
c
      implicit none
      real*8 pi
      parameter (pi=3.14159265358979d0)
      real*8 mmx,sigmag,nsig,dxe,dxs,dxa,dxb
      real*8 mmd,xc,rho,wl(10),xe,xs,xa,xb
      integer ndfp,i,nwl
      complex*16 muu(10),mu,mp(10),eps
      real*8 one,zero
      parameter (one=1.d0,zero=0.d0)
      character*1 flag
      logical dflag
c
      write(*,*) 'read(*,*) mmd,sigmag,rho,ndfp,nsig'
      read(*,*) mmd,sigmag,rho,ndfp,nsig
      write(*,*) 'read(*,*) nwl,flag,(wl(i),mp(i),i=1,nwl)'
      read(*,*) nwl,flag,(wl(i),mp(i),i=1,nwl)
      write(*,*) 'read(*,*) (muu(i),i=1,nwl)'
      read(*,*) (muu(i),i=1,nwl)
      write(*,*)
*
      dflag=rho.le.zero
      if(dflag) then
      rho=one
      write(*,1050)
      write(*,1100) mmd,sigmag,rho,ndfp,nsig
                 else
      write(*,1000) mmd,sigmag,rho,ndfp,nsig
      end if
      write(*,*)
c
      do 100 i=1,nwl
*
      if(dflag) then
      mmx=mmd
      wl(i)=one
                                    else
      mmx=pi*mmd/wl(i)
      end if
*
      if(flag.eq.'p') then
         eps=mp(i)
        else if(flag.eq.'r') then
                eps=(mp(i))**2
           else
             stop 'material properties input error'
```

```
      end if
*
      mu=muu(i)
*
      call dxtcs(mmx,sigmag,ndfp,nsig,eps,mu,dxe,dxs,dxa,dxb)
*
      xc=1.d0/wl(i)/rho
      xe=xc*dxe
      xs=xc*dxs
      xa=xc*dxa
      xb=xc*dxb
*
      write(*,1200) wl(i),xe,xs,xa,xb
      write (*,1300) eps,mu
      if(flag.eq.'r') write(*,1400) mp(i)
      write(*,*)
*
  100 continue
c
 200  stop 'done disint'
c
 1000 format('mmd = ',f5.2,' sigmag = ',f5.2,' rho = ',f5.2,
     &       ' ndfp = ',i5,' nsig = ',f5.2)
 1050 format('dimensionless extinction coefficients will be computed'/)
 1100 format('mmx = ',f5.2,' sigmag = ',f5.2,' rho = ',f5.2,
     &       ' ndfp = ',i5,' nsig = ',f5.2)
 1200 format('wl = ',f8.4,' ext = ',e10.5,' scat = ',e10.5,/13x,
     &       ' abs = ',e10.5,' bsca = ',e10.5)
 1300 format('eps = ',2f6.3,' mu = ',2f6.3)
 1400 format('refractive index = ',2f6.3)
c
      end
c
```

```
      subroutine dxtcs(mmx,sigmag,ndfp,nsig,eps,mu
     &                                          ,dxe,dxs,dxa,dxb)
c-----------------------------------------------------------------------
c>>> subroutine  dxtcs computes dimensionless extinction, scattering,   <<<
c>>> absorption, and backscatter coefficients for particulate ensembles <<<
c>>> of spheres with lognormal particle size distributions              <<<
c
c     Merrill Milham        >>> version 1.0 <<<            JANUARY 1994
c
c     Inputs:
c               mmx = mass median size parameter                    (real*8)
c             sigmag = geometric standard deviation                 (real*8)
c              ndfp = number of equally spaced (logarithmically)
c                     points at which integrand is to be
c                     evaluated                                    (integer)
c              nsig = number of geometric standard deviations on
c                     both sides of the mmx over which the integration
c                     extends                                       (real*8)
c               eps = complex permittivity of particle material (complex*16)
c                mu = complex permeability of particle material (complex*16)
c
c   outputs:
c               dxe = dimensionless extinction coefficient          (real*8)
c               dxs = dimensionless scattering coefficient          (real*8)
c               dxa = dimensionless absorption coefficient          (real*8)
c               dxb = dimensionless backscatter coefficient         (real*8)
c
c   subroutines used:
c                     simp(real*8 function ) - does Simpson's rule
c                     integration
c
c-----------------------------------------------------------------------
c
      implicit none
*
      real*8 mmx,sigmag,nsig
      integer ndfp
      complex*16 eps,mu
      real*8 dxe,dxs,dxa,dxb
*
      integer mpsd
      parameter (mpsd=3001)
      real*8 tsume(mpsd),tsums(mpsd),tsumb(mpsd),dx,dc
      real*8 lnsg,ez2x,r,xo,zo,z
      real*8 theta,qext,qsca,qbac,g
      real*8 simp
      complex*16 s1,s2
      integer i
*
      real*8 pi
      real*8 dco,sqrt2
      parameter (pi=3.14159265358979d0)
```

```
      parameter (dco=1.5d0*1.25331413731550d0,sqrt2=1.4142135623709d0)
      real*8 zero,one,two,fourpi
      parameter (zero=0.d0,one=1.d0,two=2.d0,fourpi=4.d0*pi)
c
      lnsg=dlog(sigmag)
      dc=dco/lnsg
      r=sigmag**(two*nsig/dble(ndfp-1))
      xo=mmx/(sigmag**nsig)/r
      zo=one/sqrt2/lnsg
c
   90 if(ndfp.gt.mpsd) stop 'dxtcs: arrays too small $90'
*
      theta=zero
      do 100 i=1,ndfp
      xo=xo*r
      z=zo*dlog(xo/mmx)
      ez2x=dexp(-z*z)/xo
*
      call magsph(xo,eps,mu,0,theta,qext,qsca,qbac,g,s1,s2)
*
      tsume(i)=qext*ez2x
      tsums(i)=qsca*ez2x
      tsumb(i)=qbac*ez2x
*
  100 continue
c
      dx=dlog(r)
      dxe=dc*simp(tsume,ndfp,dx)
      dxs=dc*simp(tsums,ndfp,dx)
      dxa=ddim(dxe,dxs)
      dxb=dc/fourpi*simp(tsumb,ndfp,dx)
c
      return
c
      end
```

```
      subroutine magsph(x,eps,mu,numang,theta,
     &                                  qext,qsca,qbac,g,s1,s2)
c
c  ************************************************************************
c
c  Subroutine magsph computes the scattering cross sections and angular
c  scattering from a magnetic sphere. If the number of scattering angles
c  is set to zero, only the cross sections (efficiencies) are returned.
c
c     Merrill Milham            >>> version 2.0 <<<              SEPT 1993
c
c     Inputs:
c              x = size parameter of the sphere                 (real*8)
c            eps = complex permittivity: epsr -i*epsi           (complex*16)
c             mu = complex permeability: mur - i*mui            (complex*16)
c         numang = number of scattering angles                  (integer)
c                  between 0 & 90 deg.
c          theta = scattering angles in degrees                 (real*8)
c                  theta(i) are entered between 0 & 90 deg.
c                  theta must increase monotonically. Results for
c                  supplementary angles (180 deg. - theta(i)) are
c                  also returned.
c
c     Outputs:
c            qext = extinction efficiency                       (real*8)
c            qsca = scattering efficiency                       (real*8)
c            qbac = backscatter efficiency                      (real*8)
c               g = asymmetry factor                            (real*8)
c              s1 = scattered amplitude                         (complex*16)
c              s2 = scattered amplitude                         (complex*16)
c
c     Subroutines used:
c
c             zbjy - returns one-half integer order J & Y Bessel functions
c
c     References:
c
c     M. Kerker, D.-S. Wang, and C. L. Giles, "Electromagnetic scattering
c     by magnetic spheres," J. Opt. Soc. Am., 73, 765-767 (1983).
c
c     D. E. Amos, "Algorithm 644:A portable package for Bessel functions of
c     a complex argument and nonnegative order," ACM Trans. on Math.
c     Software, 12, 265-273 (1986).
c
c     M. Abramowitz and I. A. Stegun, "Handbook of Mathematical Functions,"
c     NBS Applied Math. Series 55, US Dept. of Commerce, Washington, DC
c     (1955).
c
c     W. J. Wiscombe,"Mie Scattering Calculations: Advances in Technique and
c     Fast, Vector-Speed Computer Codes," NCAR Tech. Note, NCAR/TN-140+STR
c     (1979)
c
```

```
c  ***********************************************************************
c
      implicit none
c
      real*8 x
      complex*16 eps,mu
      integer numang
      real*8 theta(1)
*
      real*8 qext,qsca,qbac,g
      complex*16 s1(1),s2(1)
c
      integer al,nangl,nangl2
      real*8 third
      parameter (third=1.d0/3.d0,al=5100,nangl=255,nangl2=(nangl+1)/2)
      complex*16 sp(nangl2),sm(nangl2),sps(nangl2),sms(nangl2)
*
      complex*16 m,mc1,mxi,s,t,u,v,an,bn,xp
      real*8 xi,dn,dnn,rn,tnp1,thetan
      real*8 xmu(nangl),pi(nangl),pil(nangl),tau(nangl)
*
      real*8 bjr(al),byr(al)
      real*8 cjr(al),cji(al),cyr(al),cyi(al)
      real*8 bjn,byn,bjl
      real*8 sc,ca,t1,t2,t3,t4
      complex*16 cjn,cyn,cjl,cyl,h2n,h2l
      complex*16 anl,bnl,bs,anp,bnp,abp,abm,anpm,bnpm
      complex*16 zt1,zt2
*
      integer kstop,k,mm,n,j,j2,j3
*
      real*8 cpi,zero,one,two,rad,half,fnu
      complex*16 cdblei,cdble1,cdble0
      parameter (cpi=3.14159265358979323384d0,zero=0.d0,one=1.d0)
      parameter (two=2.d0,rad=cpi/180.d0,half=0.5d0,fnu=half)
      parameter (cdblei=(0.d0,1.d0),cdble1=(1.d0,0.d0))
      parameter (cdble0=(0.d0,0.d0))
c
      kstop=idint(x+4.d0*x**third+4.d0)
*
      if(kstop.le.al) then
         continue
                    else
         print*,'magsph arrays too small: kstop =',kstop,' al =',al
         stop
      end if
c
      if(numang.eq.0) then
         s1(1)=cdble0
         s2(1)=cdble0
                    else
*
```

```
            if(numang.le.nangl2) then
               continue
                                 else
           print*,numang,'scattering angles input: only',nangl2,' allowed'
               stop
            endif
*
      do 100 n=1,numang
      thetan=dabs(theta(n))
      theta(n)=thetan
*
            if(thetan.le.90.d0) then
               continue
                                else
      print*,'theta(',n,')=',thetan,'scattering angles must be < 90 deg'
            stop
            end if
*
      thetan=rad*thetan
      xmu(n)=dcos(thetan)
*
      sp(n)=cdble0
      sm(n)=cdble0
      sps(n)=cdble0
      sms(n)=cdble0
      pi(n)=half
      pil(n)=zero
*
  100 continue
*
      end if
c
      m=zsqrt(mu*eps)
      mc1=m/mu
*
      xi=one/x
      mxi=xi/m
c
      call zbjy(x,m,kstop,fnu,bjr,byr,cjr,cji)
c
      bjl=bjr(1)
      cjl=dcmplx(cjr(1),cji(1))
      cyl=dcmplx(cyr(1),cyi(1))
      h2l=dcmplx(bjr(1),-byr(1))
*
      qext=zero
      qsca=zero
      bs=cdble0
      g=zero
*
      anl=cdble0
      bnl=cdble0
```

```
      dn=one
      rn=one
      tnp1=one
      mm=1
c
      do 300 k=2,kstop
*
      tnp1=tnp1+two
      t1=dn-rn
      ca=one+rn
      sc=rn
*
      dnn=dn+one
      rn=one/dnn
*
      sc=sc+rn
*
      bjn=bjr(k)
      byn=byr(k)
      cjn=dcmplx(cjr(k),cji(k))
      cyn=dcmplx(cyr(k),cyi(k))
      h2n=dcmplx(bjn,-byn)
*
      xp=dn*mxi
      s=cj1-xp*cjn
      u=s*h2n
      s=s*bjn
*
      xp=dn*dcmplx(xi,zero)
      t=cjn*(bj1-xp*bjn)
      v=cjn*(h21-xp*h2n)
*
      an=(s-mc1*t)/(u-mc1*v)
      bn=(mc1*s-t)/(mc1*u-v)
      abp=an+bn
      abm=an-bn
*
      zt1=dconjg(an)
      zt2=dconjg(bn)
      qext=qext+tnp1*dble(abp)
      qsca=qsca+tnp1*(an*zt1+bn*zt2)
      bs=bs-(dn+half)*mm*abm
      g=g+t1*dble(an1*zt1+bn1*zt2)+sc*dble(an*zt2)
*
      if(numang.eq.0) then
         continue
                      else
         anp=sc*abp
         bnp=sc*abm
         anpm=mm*anp
         bnpm=mm*bnp
*
```

```
            do 375 j=1,numang
            t1=xmu(j)*pi(j)
            t4=t1-pil(j)
            tau(j)=dn*t4-pil(j)
            t2=pi(j)+tau(j)
            t3=pi(j)-tau(j)
*
            sp(j)=sp(j)+anp*t2
            sms(j)=sms(j)+bnpm*t2
            sm(j)=sm(j)+bnp*t3
            sps(j)=sps(j)+anpm*t3
*
            pil(j)=pi(j)
            pi(j)=t1+ca*t4
*
  375       continue
         end if
*
         dn=dnn
         mm=-mm
         an1=an
         bn1=bn
*
         bj1=bjn
         cj1=cjn
         cy1=cyn
         h21=h2n
*
  300 continue
c
         if(numang.eq.0) then
            continue
                        else
            j2=2*numang
            do 500 j=1,numang
            j3=j2-j
            s1(j)=sp(j)+sm(j)
            s2(j)=sp(j)-sm(j)
            s1(j3)=sps(j)+sms(j)
            s2(j3)=sps(j)-sms(j)
  500       continue
*
         end if
c
         xi=two*xi*xi
         qext=xi*qext
         qsca=xi*qsca
         xi=two*xi
         qbac=xi*bs*dconjg(bs)
         g=xi/qsca*g
c
         return
```

```
c
      end
```

```
      subroutine zbjy(x,m,nstop,fnu,
     &                           bjr,byr,cjr,cji)
c
c  ***********************************************************************
c
c  Subroutine zbjy gets J & Y Bessel functions for use in
c  sphere (fnu=0.50) or cylinder (fnu=0.0d0) scattering calculations.
c  Scaled or nonscaled functions for argument z = m*x are returned depending
c  on the magnitude of the product of the size parameter and the complex
c  refractive index (zabs(z)).  Nonscaled functions are returned if the
c  imaginary part of the refractive index is zero.  Nonscaled functions are
c  returned for argument x.
c
c           Merrill Milham     >>> version: 2.0 <<<            JANUARY 1994
c
c  inputs:
c             x = the size parameter of the cylinder.(real*8)
c             m = the complex refractive index, n - ik.(complex*16)
c         nstop = the highest order of the Bessel functions.(integer)
c           fnu = 0.5d0 for sphere calculations or
c                 0.0d0 for cylinder calculations. (real*8)
c
c  outputs:
c
c   bjr = real part of J(x) Bessel functions.(array: real*8)
c   byr = real part of Y(x) Bessel functions.(array: real*8)
c   cjr = real part of J(m*x) Bessel functions.(array: real*8)
c   cji = imag. part of J(m*x) Bessel functions.(array: real*8)
c
c  subroutines used:
c
c              zbesj - returns J Bessel functions
c              zbesy - returns Y Bessel functions
c
c Reference: D. E. Amos, "Algorthim 644: A Portable Package for Bessel
c            Functions of a Complex Argument and Nonnegative Order,"
c            ACM Transcations on Mathematical Software, 12,265-273(1986).
c
c***********************************************************************
c
      implicit none
      automatic cwrkr,cwrki,bji,byi
*
      integer al
      real*8 zero,xll,two
      parameter (al=5100,zero=0.0d0,xll=1.d-1,two=2.d0)
*
      real*8 bjr(1),byr(1)
      real*8 x,fnu,cjr(1),cji(1)
      real*8 zr,zi,dlmach
      real*8 cwrkr(al),cwrki(al)
      real*8 rlm5,elim,aa,alim
```

```
      complex*16 m,z
      integer kode,ierr,nz,nstop,n
      integer k1,i1mach,k2,k
      logical zflag,eflg1,eflg2
c
      kode=1
c
      call zbesj(x,zero,fnu,kode,nstop,bjr,cji,nz,ierr)
*
      eflg1=ierr.eq.0
      eflg2=nz.eq.0
      if (eflg1.and.eflg2) then
          continue
         else if (.not.eflg2.and.eflg1) then
                   nstop=nstop-nz
                   print*,'zbesj error: ierr =',ierr,'nz =',nz
                     else
              print*,'inputs =',x,zero,fnu,kode,nstop
              print*,'zbesj called from subroutine zbjy'
      end if
c
      call zbesy(x,zero,fnu,kode,nstop,byr,cji,nz,cwrkr,cwrki,ierr)
*
      eflg1=ierr.eq.0
      eflg2=nz.eq.0
      if (eflg1.and.eflg2) then
         . continue
         else if (.not.eflg2.and.eflg1) then
                   nstop=nstop-nz
                   print*,'zbesy error: ierr =',ierr,'nz =',nz
          continue
                     else
              print*,'zbesy error: ierr =',ierr,'nz =',nz
              print*,'inputs =',x,zero,fnu,kode,nstop
              print*,'zbesy called from subroutine zbjy'
      end if
c
      z=m*x
      zr=dble(z)
      zi=dimag(z)
      if(zi.ne.zero) then
*
      k1 = i1mach(15)
      k2 = i1mach(16)
      r1m5 = d1mach(5)
      k = min0(iabs(k1),iabs(k2))
      elim = 2.303d0*(dble(float(k))*r1m5-3.0d0)
      k1 = i1mach(14) - 1
      aa = r1m5*dble(float(k1))
      aa = aa*2.303d0
      alim = elim + dmax1(-aa,-41.45d0)
*
```

```
         if(dabs(zi).gt.alim) kode=2
                                        else
         continue
       end if
c
       call zbesj(zr,zi,fnu,kode,nstop,cjr,cji,nz,ierr)
*
       eflg1=ierr.eq.0
       eflg2=nz.eq.0
       if (eflg1.and.eflg2) then
           continue
          else if (.not.eflg2.and.eflg1) then
                    nstop=nstop-nz
                    print*,'zbesj error: ierr =',ierr,'nz =',nz
           continue
                     else
               print*,'zbesj error: ierr =',ierr,'nz =',nz
               print*,'inputs =',zr,zi,fnu,kode,nstop
               print*,'zbesj called from subroutine zbjy'
       end if
c
       zflag=dabs(zi).lt.two*d1mach(1).and.x.lt.xll
       if (.not.zflag) then
            continue
                     else
            do 100 n=1,nstop
               cji(n)=zero
  100       continue
       end if
*
       zflag=dabs(zr).lt.two*d1mach(1).and.x.lt.xll
       if (.not.zflag) then
            continue
                     else
            cji(1)=zero
            do 200 n=2,nstop
                   if(mod(n,2).eq.0) then
                      cjr(n)=zero
                                     else
                      cji(n)=zero
                   end if
  200       continue
       end if
c
       return
c
       end
```

```
      real*8 function simp (y,n,dx)
c*****************************************************************
c     function simp does simpson's rule integration
c     inputs:
c             y = array containing ordinate values      (real*8)
c             n = number of ordinate values             (integer)
c            dx = abscissa increment(constant) between
c                 ordinate values                      (real*8)
c     output:
c          simp = estimated value of the integral       (real*8)
c
c*****************************************************************
c
      implicit none
      real*8 s1,s2,s4,third
      real*8 dx,y(1)
      integer n,ne,i
      parameter (third=1.d0/3.d0)
c
      if(n.lt.5) then
      write(6,1000)
 1000 format(' error function : simp')
      write(6,'('' number of integration points must be .ge. 5 n = ''
     +     ,i10)') n
      stop
c
      else if(mod(n,2) .ne.1) then
      write(6,1000)
      write(6,'('' number of integration points must be odd. n = '',i10)
     +     ') n
      stop
c
      else
      continue
      end if
c
      ne=n-3
      s1=y(1)+y(n)
      s2=0.d0
      s4=y(n-1)
c
      do 100 i=2,ne,2
      s4=s4+y(i)
      s2=s2+y(i+1)
  100 continue
c
      s4=4.d0*s4
      s2=2.d0*s2
c
      simp=third*dx*(s1+s2+s4)
c
      return
```

Blank

APPENDIX B: SAMPLE OUTPUT

```
                                        INPUT

10,1.01,-1,151,5
1,'r',1,(1.5,-.1)
(1,0)

                                        OUTPUT


 read(*,*) mmd,sigmag,rho,ndfp,nsig
 read(*,*) nwl,flag,(wl(i),mp(i),i=1,nwl)
 read(*,*) (muu(i),i=1,nwl)

dimensionless extinction coefficients will be computed

mmx = 10.00 sigmag =  1.01 rho =  1.00 ndfp =   151 nsig =  5.00

wl =   1.0000 ext = .11589E+01 scat = .58179E+00
              abs = .57709E+00 bsca = .33692E-02
eps =  2.240-0.300 mu =  1.000 0.000
refractive index =  1.500-0.100

                                        INPUT

10,1.4,-1,151,5
1,'r',1,(1.5,-.1)
(1,0)

                                        OUTPUT


 read(*,*) mmd,sigmag,rho,ndfp,nsig
 read(*,*) nwl,flag,(wl(i),mp(i),i=1,nwl)
 read(*,*) (muu(i),i=1,nwl)

dimensionless extinction coefficients will be computed

mmx = 10.00 sigmag =  1.40 rho =  1.00 ndfp =   151 nsig =  5.00

wl =   1.0000 ext = .12361E+01 scat = .63179E+00
              abs = .60428E+00 bsca = .35034E-02
eps =  2.240-0.300 mu =  1.000 0.000
refractive index =  1.500-0.100

                                        INPUT


10,1.4,-1,151,5
1,'p',1,(2.24,-.3)
(2.24,-.3)

                                        OUTPUT
```

```
 read(*,*) mmd,sigmag,rho,ndfp,nsig
 read(*,*) nwl,flag,(wl(i),mp(i),i=1,nwl)
 read(*,*) (muu(i),i=1,nwl)

dimensionless extinction coefficients will be computed

mmx = 10.00 sigmag =  1.40 rho =  1.00 ndfp =   151 nsig =  5.00

wl =   1.0000 ext = .12292E+01 scat = .57928E+00
              abs = .64991E+00 bsca = .42778E-32
eps =  2.240-0.300 mu =  2.240-0.300
```